Published in final edited form as:
Comput Methods Programs Biomed. 2009 August ; 95(2): 95–104. doi:10.1016/j.cmpb.2009.01.009.

An Event-Driven Distributed Processing Architecture for Image-Guided Cardiac Ablation Therapy

M.E. Rettmann, **D.R. Holmes III**, **B.M. Cameron**, and **R.A. Robb**
Biomedical Imaging Resource, Mayo Clinic College of Medicine, Rochester, MN, USA

Abstract

Medical imaging data is becoming increasing valuable in interventional medicine, not only for preoperative planning, but also for real-time guidance during clinical procedures. Three key components necessary for image-guided intervention are real-time tracking of the surgical instrument, aligning the real-world patient space with image-space, and creating a meaningful display that integrates the tracked instrument and patient data. Issues to consider when developing image-guided intervention systems include the communication scheme, the ability to distribute CPU intensive tasks, and flexibility to allow for new technologies. In this work, we have designed a communication architecture for use in image-guided catheter ablation therapy. Communication between the system components is through a database which contains an event queue and auxiliary data tables. The communication scheme is unique in that each system component is responsible for querying and responding to relevant events from the centralized database queue. An advantage of the architecture is the flexibility to add new system components without affecting existing software code. In addition, the architecture is intrinsically distributed, in that components can run on different CPU boxes, and even different operating systems. We refer to this **F**ramework for **I**mage-Guided **N**avigation using a **D**istributed **E**vent-Driven Database in **R**eal-Time as the FINDER architecture. This architecture has been implemented for the specific application of image-guided cardiac ablation therapy. We describe our prototype image-guidance sytem and demonstrate its functionality by emulating a cardiac ablation procedure with a patient-specific phantom. The proposed architecture, designed to be modular, flexible, and intuitive, is a key step towards our goal of developing a complete system for visualization and targeting in image-guided cardiac ablation procedures.

1 Introduction

Image guidance for interventional procedures is becoming increasingly valuable in clinical practice. While preoperative datasets are important for planning purposes, their utility is even greater when integrated into the procedure room and used directly for guidance. Preoperative imaging data can captures patient-specific function, detailed anatomy, or a combination of function fused with anatomy. The imaging data can be used directly, or post-processing algorithms may be applied to extract relevant information for the specific application. For example, relevant anatomical structures can be segmented from the data and represented as a computer model, often a triangulated surface mesh.

Conflict of Interest Statement: R.A. Robb and Mayo Clinic have a financial interest in technology used in this research and have received royalties greater than the federal threshold for significant financial interest in the preceding 12 months from licensing this technology.

Publisher's Disclaimer: This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

There are several key components necessary in order to integrate patient-specific imaging data into a surgical procedure (for a complete review see [1]). First, the surgical instrument must be tracked in three-dimensional space throughout the procedure. Second, a mapping must be computed in order to register the real-world space of the patient into the image-space of the preoperative data. This often involves identifying corresponding anatomical locations between the patient-specific image data and the physical anatomy of the patient in the procedure room. Anatomical points in image-space can be identified directly in the image data, or on a model generated from post-processing analysis, such as a surface representation of segmented anatomy. The corresponding locations in real-world patient space are located using a tracked device. After a collection of landmark pairs have been identified, a point-matching registration is computed to bring the real-world patient space into alignment with the preoperative data. The third key component for integration of preoperative data is the ability to display the tracked surgical device within the patient-specific data in a way that is useful to the clinician.

Several challenges exist when developing systems for image-guided intervention. The first involves communication between the system components. Various data streams, including information from tracked devices, updated registration information, as well as user input, must be available and shared between all system components. Another challenge is that guidance systems must be designed such that tasks requiring large CPU usage can easily be distributed in order to avoid placing a computational burden on any single processor. Finally, the architecture must be flexible to allow for facile integration of new devices as they become available.

In order to address these complexities, previous work in constructing image-guided systems have utilized modular designs in which different functionalities are implemented as separate components. For example, the ORION PC-based system [2], implements the different system components (tracker, registration, visualization), as separate dynamically linked libraries. Communication between components is coordinated by a message pump that sends and receives messages across the system. Another component-based architecture is a collection of open source C++ classes specifically designed for image-guided procedures [3,4]. Communication is accomplished by pushing and pulling data between system components and using state machines to insure valid, deterministic states at all times.

In this work, we have designed a communication architecture for use in our prototype image-guided cardiac ablation system. As in previous work [2,3,4], our design incorporates a modular, component-based architecture, however, the communication scheme is unique in that each system component is responsible for querying and responding to relevant events from a centralized database queue. The architecture is flexible in that new system components can be added without affecting existing software code. The design is also intrinsically distributed, without the need for any specialized software. We refer to this **F**ramework for **I**mage-Guided **N**avigation using a **D**istributed **E**vent-Driven Database in **R**eal-Time as the FINDER architecture. In this paper, we provide a detailed description of this event-driven communication architecture and demonstrate its functionality by emulating an ablation procedure using a patient-specific physical phantom. We also describe how this framework could be extended for use in other image-guidance systems.

2 Methodologies

2.1 Overall Design

The generic FINDER architecture is shown in Figure 1(a). Communication between the various system components is through a centralized database, as represented in the center of the figure. The database contains both an event queue as well as data tables and supports the communication between the visualization component, the acquisition servers, and the processing servers. The specific implementation of this architecture for image-guided cardiac ablation therapy is shown in Figure 1(b). Visualization is accomplished through a user interface which displays the patient-specific cardiac model along with the tracked catheter. The point tracking acquisition server is connected to a tracked catheter and is responsible for providing both real-time updates on three-dimensional location, as well as specific points collected for the registration procedure. The registration server computes a registration between real-world patient space and image-space. As events are generated by either the interface or the servers they are added to the event queue maintained by the database. The interface and servers each contain event processing loops, which query and process relevant events from the database queue. In the next sections, we describe each system component in detail.

2.2 Event Queue

Communication between system components occurs through an event queue which is maintained as a table in the centralized database. A brief description of all possible event types is given in Figure 2.

The *Image Data* event, generated by the user interface, indicates that new data has been loaded. All system components respond by entering a *live* state. The *Configuration* event, also generated by the user interface, indicates that a configuration setting changed, such as the registration type. All system components respond by changing relevant configuration settings. The *Point* event, generated by either the point tracking server or the interface, indicates that a point has been added to the database. The interface responds by drawing this point into the visualization and the registration server computes an updated registration. The architecture is general enough to handle any number of different point types, a detailed description of the managament of different point types is given in Section 2.3.

The *Registration* event is generated by the registration server and indicates that a new registration has been computed. The user interface responds to this event by incorporating the new transformation matrix into the visualization. The *Link* event is generated by the user interface and indicates that a new landmark pair has been defined. One point is a location in real-world space and the other point is its corresponding location in image-space. The interface updates the visualization by attaching corresponding labels to the identified landmark pair. The registration server responds to this event by incorporating the matched set of points into the registration algorithm. The *Point Status* event is generated by the user interface and indicates that a point has either been deleted or undeleted. The user interface updates the visualization and the registration server computes a new registration matrix.

The *Stop* event is generated by the user interface and indicates that the current procedure has ended. The event is generated when either the current case is closed or a new case is opened. All system components respond by altering their state to no longer *live*. The *Quit* event is generated by the interface and indicates that all programs should exit. The event is generated when the user exits the interface. All system components responding by closing.

2.3 Database Design

In this section, we describe the details of the database design, including the database tables and their dependencies. Figure 3 provides a graphical depiction of the overall design. The **Event** table stores the event queue. Each event has a primary key, a timestamp, and an event type (see Table 2 for a description of event types). In addition, for all events except *Stop* or *Quit*, an entry is also made into a second table which contains the details of that event. The “Table Entry ID” is the primary key of the entry into the second table. For example, when a point is inserted into the database, a *Point* event is inserted into the **Event** table, and a second entry is made into the **Point** table which contains the detailed point information.

The **Image Data** table stores information associated with the *Image Data* events. Each entry into this table contains a primary key, a filename associated with the preoperative data, and (optionally), a filename associated with a computer model of the data, such as a surface mesh.

The **Point** table stores information associated with the *Point* events. Each entry in this table contains a primary key, a point type, the three-dimensional point location (x,y,z), and a deleted flag. The deleted flag indicates if this point has been deleted via the user interface. The point tracking server is triggered to begin collecting points after receiving an *Image Data* event, thus, all collected points are associated with a specific entry in the **Image Data** table. The “Image Data ID” field contains the primary key of this entry, thereby associating all points with the appropriate patient case. Equivalently, this field could have referenced the primary key in the **Event** table, however, from a user standpoint, it was easier to attach a label that directly linked to the filename of the patient data.

The **Linkage** table stores information associated with the *Link* events. Each entry into this table contains the event IDs for the matched points. The **Point Type** table manages the different types of points relevant to the specific application. Points can be defined either in image space or in patient space. If defined in image-space, the points do not need to undergo a transformation in order to align with the anatomical computer model. If defined in patient-space, the transformation matrix must first be applied to the point location in order to bring it into alignment with the image-space. The “Apply Registration” flag is used to indicate if a transformation is required.

The **Configuration** table records the current registration type for use in the registration server. The **Registration** table stores the current transformation matrix, the error associated with the registration, and the number of points currently used in the registration. The **Continuous Tracked Point** table stores information about the continuously tracked catheter. Each entry in this table contains the three-dimensional location of the tracked surgical instrument (x,y,z) and its orientation (a,b,c). Since this table is continuosly updated with the most recent location of the tracked instrument, an event is not generated for each entry into this table. Instead, the user interface checks this table whenever it is not processing another event.

All communication with the database occurs through a middleware database utility library, thereby eliminating the need for programs to make direct queries to or from the database. The database utility library is implemented in the C programming language, and compiled into both a linux and windows version. The interface and servers dynamically link the approriate database utility library.

A benchmark analysis was conducted to evaluate latency of the proposed architecture. The analysis assessed the amount of time required to insert and retrieve a continuous tracked point from the database. Each table entry comprises 9 fields for a total of 36 bytes. 100,000 points were inserted and retrieved from the database with an average latency of 0.316 milliseconds per table entry point.

2.4 User Interface and Servers

State diagrams for the user interface and the processing servers are given in Figures 4 through 6. In each diagram, Read() indicates an event read from the database queue and Generate() indicates an event is generated.

The state diagram for the user interface is shown in Figure 4. The primary task of the user interface is to provide an integrated display of the preoperative image data with the tracked surgical instrument. A procedure is initiated when the user opens a new patient case from the interface. Next, the interface enters the Live state in which events are queried from the database and user events are responded to. For example, if the user selects a landmark location point on the interface, the interface inserts this point into the database, concurrently generating a *Point* event. The interface responds to this *Point* event by adding it to the display with the specified physical properties (color, transparency, etc.). Whenever the interface is not processing an event, it reads from the **Continuous Tracked Point** table to update the displayed location of the tracked instrument.

An important aspect of the interface is that it provides the ability for the user to interact with the preoperative image-data, the computer model, and points associated with the procedure. Of particular importance is the ability to define point pairs that link locations in image-space with points in patient-space for the purpose of landmark registration. A mouse click is used to identify an anatomical location in image-space and the tracked instrument is used to identify the corresponding location in patient-space. The points are then linked together via the user interface. The interface can also be used to delete points that were erroneously added, or restore previously deleted points.

The state diagram for the point tracking server is shown in Figure 5. Upon receipt of an *Image Data* event, the point tracking server enters into the Live mode, and begins inserting points into the database. Two types of interrupts can occur from the tracked catheter system. The first is a point interrupt which indicates a specific point has been collected for use in the registration algorithm. The second interrupt is for continuous motion tracking.

The state diagram for the registration server is shown in Figure 6. Following an image load, the registration server responds to *Point* and *Link* events by appropriately incorporating them into the registration when the minimum number of points or landmark pairs have been obtained. *Configuration* events can be used to alter the type of registration algorithm utilized. *Stop* and *Quit* events are used to leave Live mode and close the program.

In the event of a system failure, a recovery operation can be used to resume the system state for the most recent case. The recovery operation is initiated from the user interface and operates through a function call to the database library. The recovery function reinserts all events since the last *Image Data* event into the database queue. Since all processes operate in a modular event-driven fashion, it is not necessary to have any special recovery mode, instead, events are simply reprocessed by the servers.

3 Emulation of Image-Guided Cardiac Ablation Procedure

Atrial fibrillation is a condition in which the atria of the heart beat rapidly and irregulary. While the exact disease mechanism is not yet fully understood, it is believed that many cases are the result of aberrant electrical signals originating in the pulmonary veins [5]. This condition is often treated pharmacologically; however, drawbacks of drug therapy can include undesirable side effects and patient unresponsiveness [6]. This has motivated the development of other treatment strategies, including cardiac catheter ablation therapy. In this technique, radiofrequency energy is delivered via a catheter to the endocardial surface producing lesions that block aberrant electrical pathways, thereby eliminating arrhythmias.

A standard treatment strategy is to isolate the pulmonary veins from the left atrium by applying radiofrequency energy in a figure-eight pattern around each pair of veins. Thus, effective visualization of the target tissue is a key component for successful procedure outcomes. Standard visualization tools available to the cardiologist include 2D ultrasound and bi-plane fluoroscopy. In recent years, there has been increased interest in incorporating detailed, patient-specific anatomy into the ablation procedure [7,8,9,10,11]. In this section, we describe our prototype system for image-guided cardiac ablation therapy using the proposed FINDER architecture. Functionality of the system is demonstrated by emulating an ablation procedure using a patient-specific phantom in our laboratory.

3.1 System Description

A photograph of the physical system is shown in Figure 7. The CPU box contains dual 2.4 GHz Athlons with 2 GB of memory and runs the CentOS Distribution of Linux. The interface, point tracking server, and registration servers all run on this box. The point tracking server interfaces to a commercial EP mapping system (Biosense Webster Inc., Diamond Bar, CA) which transmits locations of the tracked catheter to our system through an interrupt-driven ActiveX component provided by Biosense Webster. Since the programming interface for the commercial tracking system is Windows based, the point tracking server was developed under Windows XP, and links with the Windows version of the database utility libraries. The executable is subsequently run on the Linux box using the open source software Wine [12], which is a Windows API for Unix.

The commercial mapping system uses a magnetic tracking technology [13], with the reference catheter typically affixed to the back of the patient. A second catheter is used for mapping and ablating the left atrium. In addition to the continuous tracked locations, specific points can be recorded by positioning the catheter at a specific location and using a foot pedal or mouse click to record the 3D location. These points are located on the endocardial surface of the left atrium and used in the landmark and surface based registration algorithms. The last component of the system is a burn switch which is turned on during ablation and used to track points collected during application of RF energy.

A snapshot of the user interface developed with AVW libraries, OpenInventor, and QT is shown in Figure 8. The display includes a volume rendering of the preoperative image data merged with a surface model of the left atrium and pulmonary veins. In order to construct the surface model, the left atrium and pulmonary veins are segmented from a preoperative CT dataset using the Analyze software [14]. Next, a surface tiling algorithm [15] is used to generate a triangular mesh representation of the anatomy. The current position and orientation of the catheter is represented as a purple cone. Different types of points are represented as colored spheres or cubes. In addition to the main integrated display, the interface provides individual data visualization windows along the left side of the main window. In these windows, the data is displayed in various formats including cross-sectional (top), volumetric (bottom), and a surface model (center).

In our prototype system, there are four types of points. The first is *Sampled* which is a point on the endocardial surface sampled by the tracked catheter. These points are linked with the corresponding location on the surface model for landmark matching or used as generic points in a point-to-surface based registration algorithm. The second type is *Burn* which is a point sampled while the burn switch is turned on. The third type is *Manual* which is a user selected point on the surface model. A *Manual* point is linked to a *Sampled* point for landmark-based registration. The final type is *Target* which is a point inserted through the user interface indicating an interventional target location determined from preoperative planning.

The registration server supports three types of registration methods. The first is landmark-based which requires a minimum of three landmark pairs and uses a least-squares fit minimization. The second is a surface based-registration and requires a minimum of five surface points. A center of mass alignment is used for initialization followed by a minimization of the point-to-surface distance [16]. The third registration option, which is the default, uses the landmark-based approach for initialization followed by refinement with the surface-based algorithm.

3.2 Procedure Emulation

For the emulation, we constructed a patient-specific physical phantom of the left atrium and pulmonary veins. The steps for the phantom construction are as follows. First, the blood pool of the left and right atria, superior and inferior vena cava, and pulmonary veins was segmented from a patient preoperative CT data set using the Analyze software [14]. The segmentation was used to create a surface model that was physically constructed using a 3D printer. Fiducial markers were identified in the image data and marked on the physical model during printing thereby providing direct correspondence between the image-space and real-world physical space [17].

In order to register real-world space with image-space the magnetically tracked catheter is used to collect anatomical landmarks in real-world space. The same anatomical locations are identified on a surface model of the left atrium and a least squares fit between the set of points is computed. This procedure is illustrated in Figure 9. The landmarks identified on the surface are shown as green cubes and the landmarks in real-world physical space as yellow spheres. Points are shown before and after alignment in Figures 9(a) and (b) respectively. The magenta cone represents the location of the catheter tip which is transformed using the same registration matrix. As additional surface points are collected with the tracked catheter, the registration is refined using the point-to-surface registration as demonstrated in Figure 9(c). Target points, corresponding to the phantom fiducial markers, are represented are orange cubes as shown in Figure 9(d).

Targeting the correct anatomical locations is a key component of the cardiac ablation procedure. In order to demonstrate targeting of predefined locations, we utilize one of the phantom fiducials which is defined on both the computer model and the physical phantom. First, the catheter is positioned at the fiducial location in physical space, as illustrated in Figure 9(e). If a correct transformation has been computed, the catheter tip in the visualization should be located at the corresonding fiducial on the surface model. The actual display is shown in Figure 9(f), providing a qualitative verification of the registration. In order to emulate a burn, the catheter is navigated to the ostium of the left superior pulmonary vein on the physical model and the burn switch is turned on. Points collected during the emulated burn are colored in red as shown in Figure 9(g). Figure 9(h) illustrates a different view of the burn pattern is which part of the surface is “cut-away” to obtain a view of the ablation from inside the left atrium.

4 Discussion

In this work, we describe a framework for image-guided navigation using a distributed event-driven real-time database, which we refer to as the FINDER architecture. Each system server runs in an autonomous fashion, making the design intrinsically distributed, since each server can potentially run on a separate CPU. Once the infrastructure is in place, it is straightforward to integrate new components into the system, using simple C-code function calls from the database utility library. All communication with the database and management of database structures are maintained by the library, and thereby, are invisible to the user. Incidentally, even the particular database implementation could be modified, and only the database utility library would need to be updated.

We have demonstrated this architecture for the specific application of cardiac ablation therapy by developing a prototype system which integrates with the Biosense Webster commercial mapping system. The architecture, however, is general enough to interface with any commercial EP mapping system, provided a programming interface for acquiring the location of the tracked catheter points is available. These tracked locations can then be inserted into the database and utilized by other system components without any changes to the existing software. The design could also be extended for use in other image-guided interventions. Any tracked instrument could be used in place of the tracked catheter, in fact, the framework is general enough to incorporate multiple tracked instruments by adding a database table for each additional instrument. Multiple registration matrices could then be handled with the same scheme used to support multiple point types.

In conclusion, we have proposed a novel framework for image-guidance and demonstrated its functionality on our prototype cardiac ablation system. The overall goal is to provide effective visualization of patient specific anatomy leading to accurate targeting for image-guided therapy. We believe the proposed system architecture, designed to be modular, flexible, and intuitive, is a key step towards this goal.

Acknowledgments

The authors thank Dr. Douglas Packer and Jon Camp for their contributions to this work. This research was supported by NIH grant RO1EB002834 from the National Institute of Biomedical Imaging and Bioengineering.

References

1. G R Jr. The process and development of image-guided procedures. Ann Rev Biomed Eng 2001;3:83–108. [PubMed: 11447058]

2. Stefansic J, Bass W, Hartmann S, Beasley R, Sinha T, Cash D, Herline A, G R Jr. Design and implementation of a PC-based image-guided surgical system. Computer Methods and Programs in Biomedicine 2002;69:211–224. [PubMed: 12204449]

3. Gary K, Ibanez L, Aylward S, Gobbi D, Blake MB, Cleary K. IGSTK: An open source software toolkit for image-guided surgery. Computer 2006:46–53.

4. Enquobahrie A, Cheng P, Gary K, Ibanez L, Gobbi D, Lindseth F, Yaniv Z, Aylward S, Jomier J, Cleary K. The image-guided surgery toolkit igstk: an open source C++ software toolkit. Journal of Digital Imaging 2007;20:21–33. [PubMed: 17703338]

5. Blaauw Y, Crijns H. Atrial fibrillation: insights from clinical trials and novel treatment options. Journal of Internal Medicine 2007;262:593–614. [PubMed: 18028181]

6. Packer D, Asirvatham S, Munger T. Progress in nonpharmacologic therapy of atrial fibrillation. Cardiovasc Electrophysiol 2003;14:S296–309.

7. Dickfeld T, Calkins H, Zviman M, Kato R, Meininger G, Lickfett L, Berger R, Halperin H, Solomon S. Anatomic stereotactic catheter ablation on three-dimensional magnetic resonance images in real time. Circulation 2003;108:2407–2413. [PubMed: 14568905]

8. Lacomis J, Wigginton W, Fuhrman C, Schwartzman D, Armfield DR, Pealer K. Multi-detector row CT of the left atrium and pulmonary veins before radio-frequency catheter ablation for atrial fibrillation. RadioGraphics 2003;23:S35–50. [PubMed: 14557500]

9. Mansour M, Holmvang G, Ruskin J. Role of imaging techniques in preparation for catheter ablation of atrial fibrillation. J Cardiovasc Electrophysiol 2004;15:1107–1108. [PubMed: 15363090]

10. Reddy V, Malchano Z, Holmvang G, Schmidt E, d'Avila A, Houghtaling C, Chan R, Ruskin J. Integration of cardiac magnetic resonance imaging with three-dimensional electroanatomic mapping to guide left ventricular catheter manipulation. Journal of the American College of Cardiology 2004;44(11):2202–13. [PubMed: 15582319]

11. Sun, Y.; Azar, F.; Xu, C.; Hayam, G.; Preiss, A.; Rahn, N.; Sauer, F. Registration of high-resolution 3D atrial images with elecroanatomical cardiac mapping: Evaluation of registration methodology; SPIE Medical Imaging; 2005. p. 299-307.

12. http://www.winehq.org/

13. Gepstein L, Hayam G, Ben-Haim S. A novel method for nonfluoroscopic catheter-based electroanatomical mapping of the heart: In vitro and in vivo accuracy results. Circulation 1997;95 (6):1611–1622. [PubMed: 9118532]

14. Robb, R.; Hanson, D. Analyze: A software system for biomedical image analysis. First Conference on Visualization in Biomedical Computing; 1990. p. 507-518.

15. Lin, W.; Robb, R. Dynamic volume texture mapping and model deformation for visually realistic surgical simulation. Proceedings of Medicine Meets Virtual Reality; 1999. p. 198-204.

16. Su, Y.; H, D., III; Rettmann, M.; Robb, R. A piecewise function-to-structural registration algorithm for image guided cardiac catheter ablation; SPIE Medical Imaging; 2006.

17. H, D., III; Rettmann, M.; Cameron, B.; Camp, J.; Robb, R. Developing patient-specific anatomic models for validation of cardiac ablation guidance procedures; SPIE Medical Imaging; 2008.

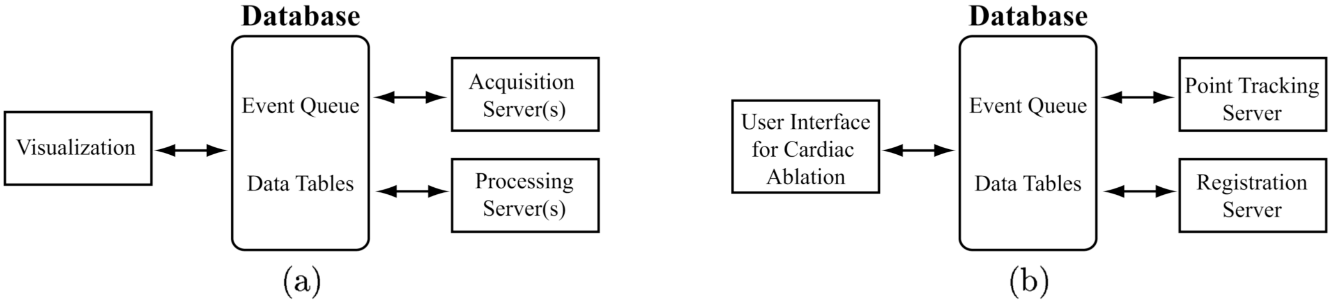
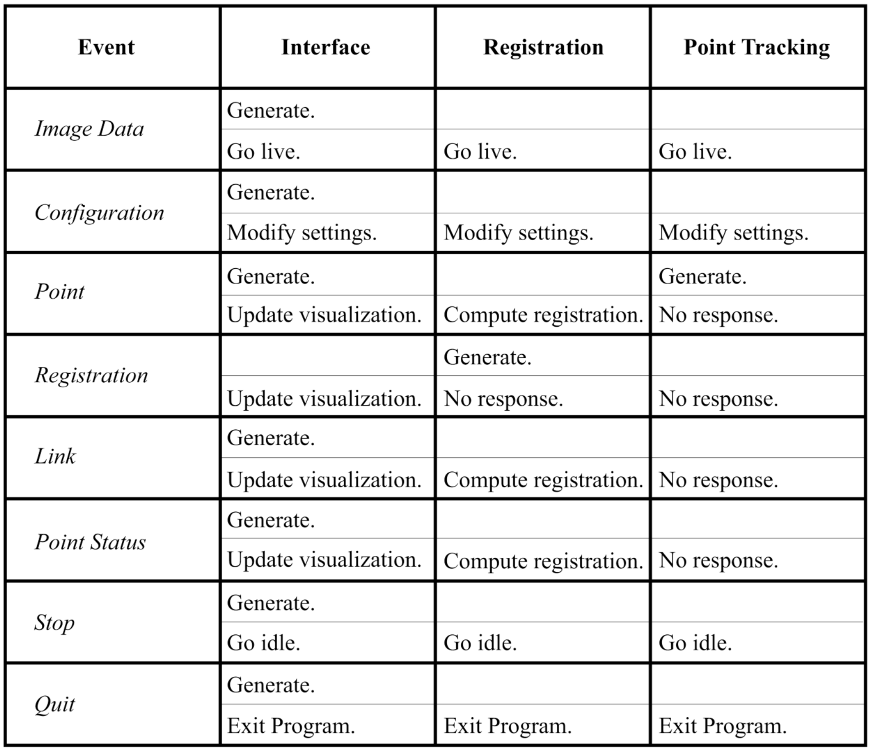


Figure 1.
(a) Generic FINDER architecture. Communication between system components takes place through an event queue located in a centralized database. (b) Specific architecture for application of cardiac ablation image-guidance.



Event	Interface	Registration	Point Tracking
Image Data	Generate.		
	Go live.	Go live.	Go live.
Configuration	Generate.		
	Modify settings.	Modify settings.	Modify settings.
Point	Generate.		Generate.
	Update visualization.	Compute registration.	No response.
Registration		Generate.	
	Update visualization.	No response.	No response.
Link	Generate.		
	Update visualization.	Compute registration.	No response.
Point Status	Generate.		
	Update visualization.	Compute registration.	No response.
Stop	Generate.		
	Go idle.	Go idle.	Go idle.
Quit	Generate.		
	Exit Program.	Exit Program.	Exit Program.

Figure 2.

Types of system events. The top portion of each table cell indicates if a system component generates an event and the bottom portion provides a brief description of the response to an event.

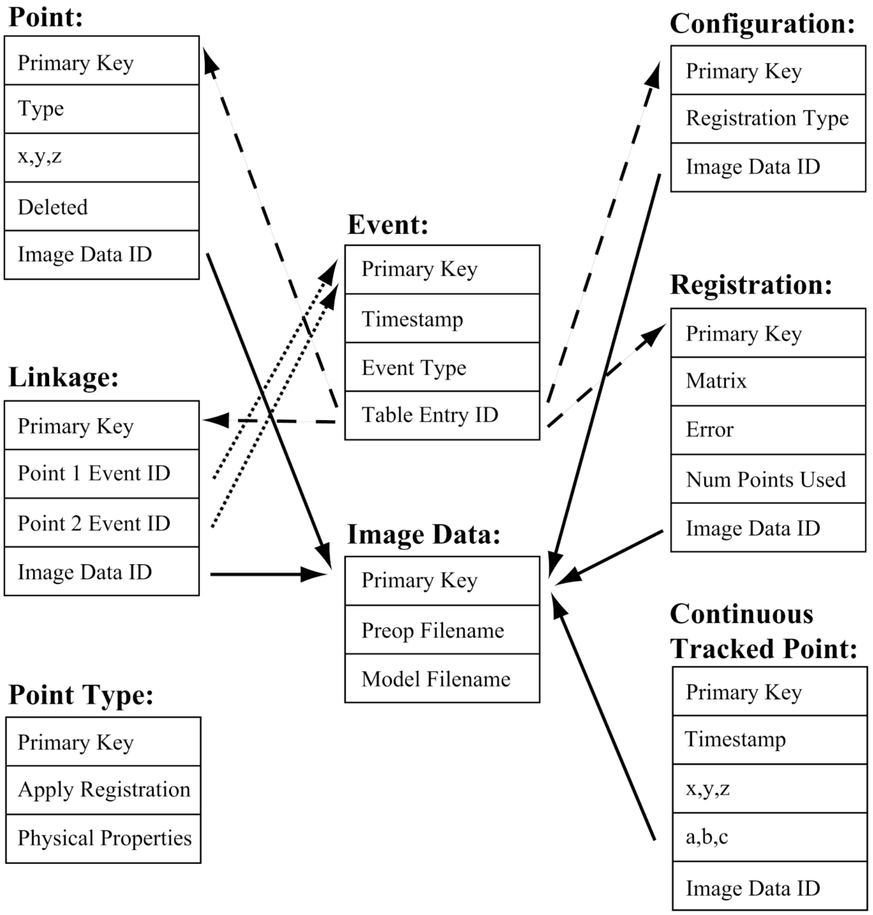


Figure 3.

Database tables and their dependencies. Solid arrows point from data tables to the associated data event. Dashed errors point from the event table to the corresponding entry in a data table. Dotted errors associate points in the Linkage table with their events in the Event table.

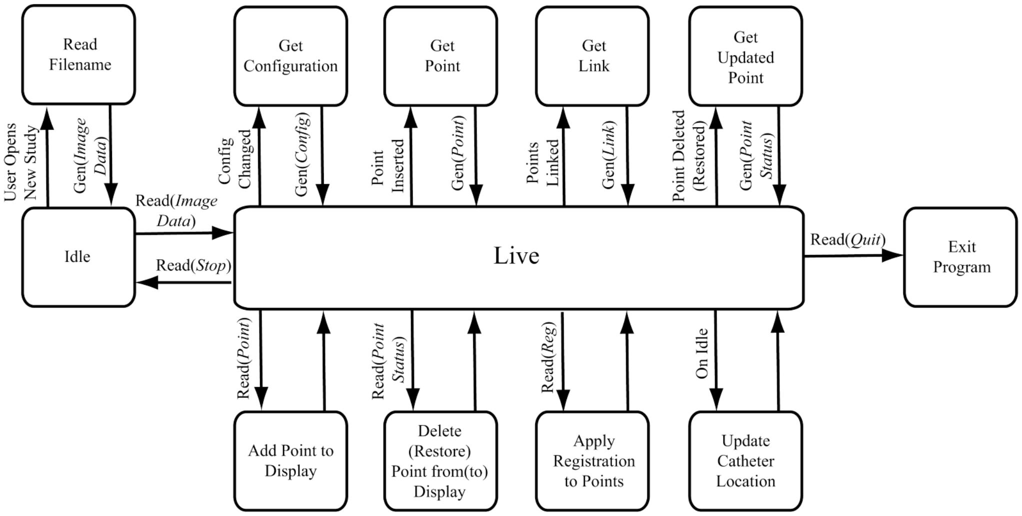


Figure 4.

State diagram for user interface.

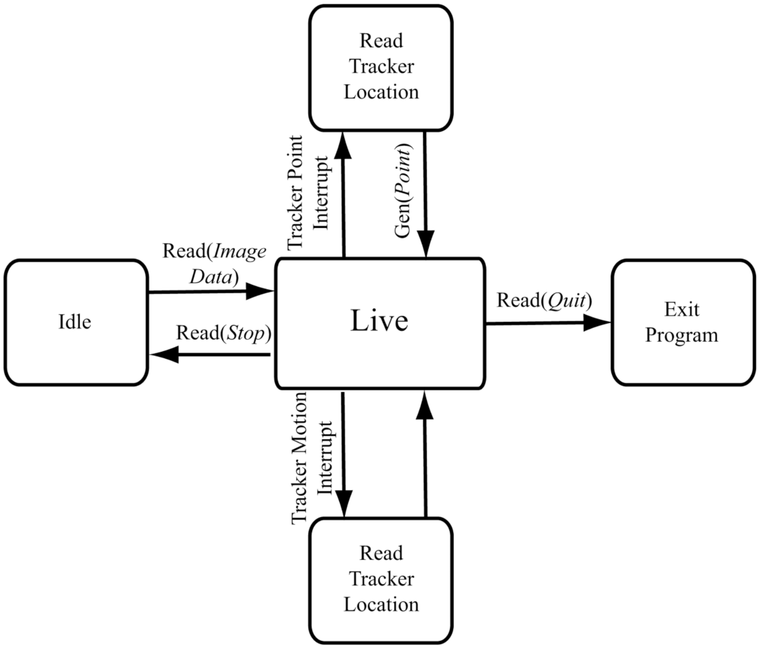


Figure 5.
State diagram for point tracking server.

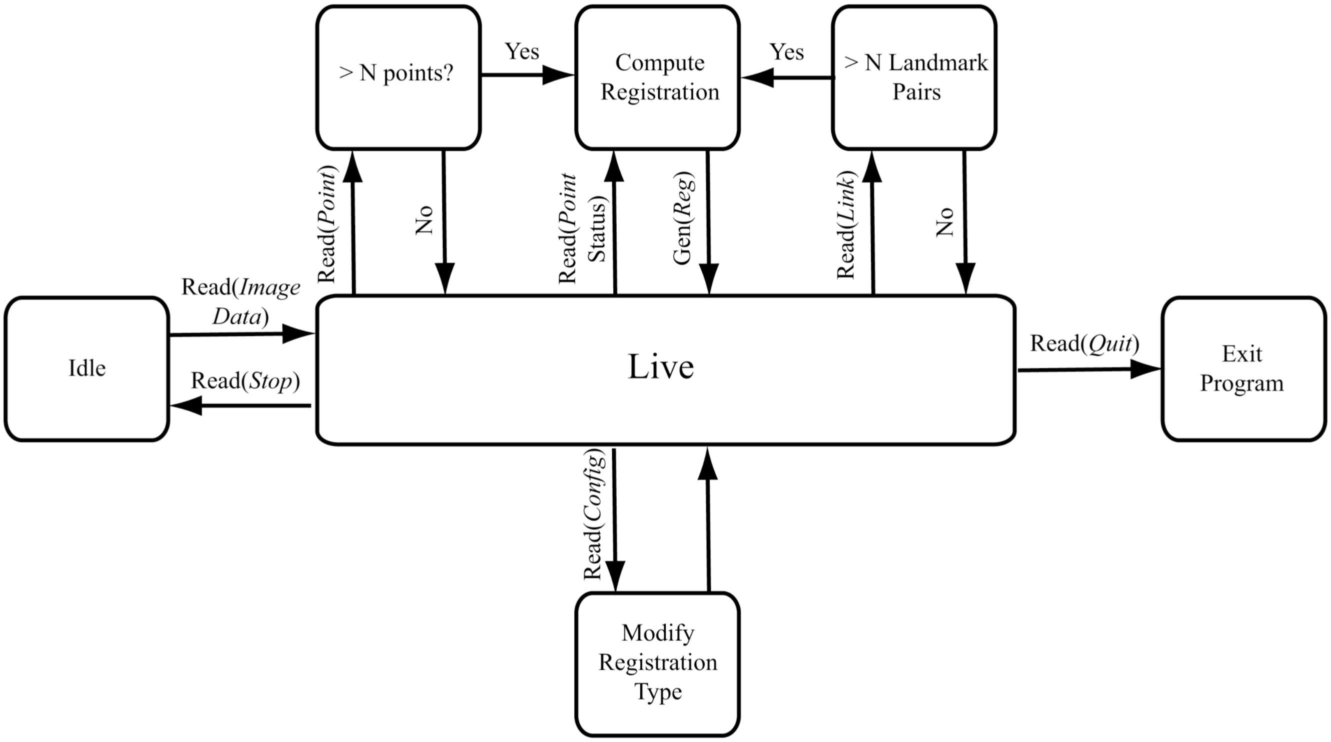


Figure 6.
State diagram for registration server.

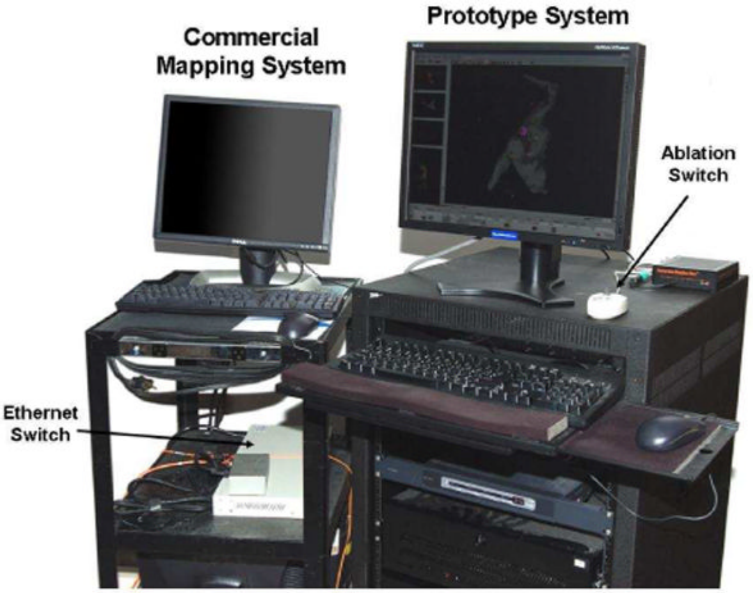
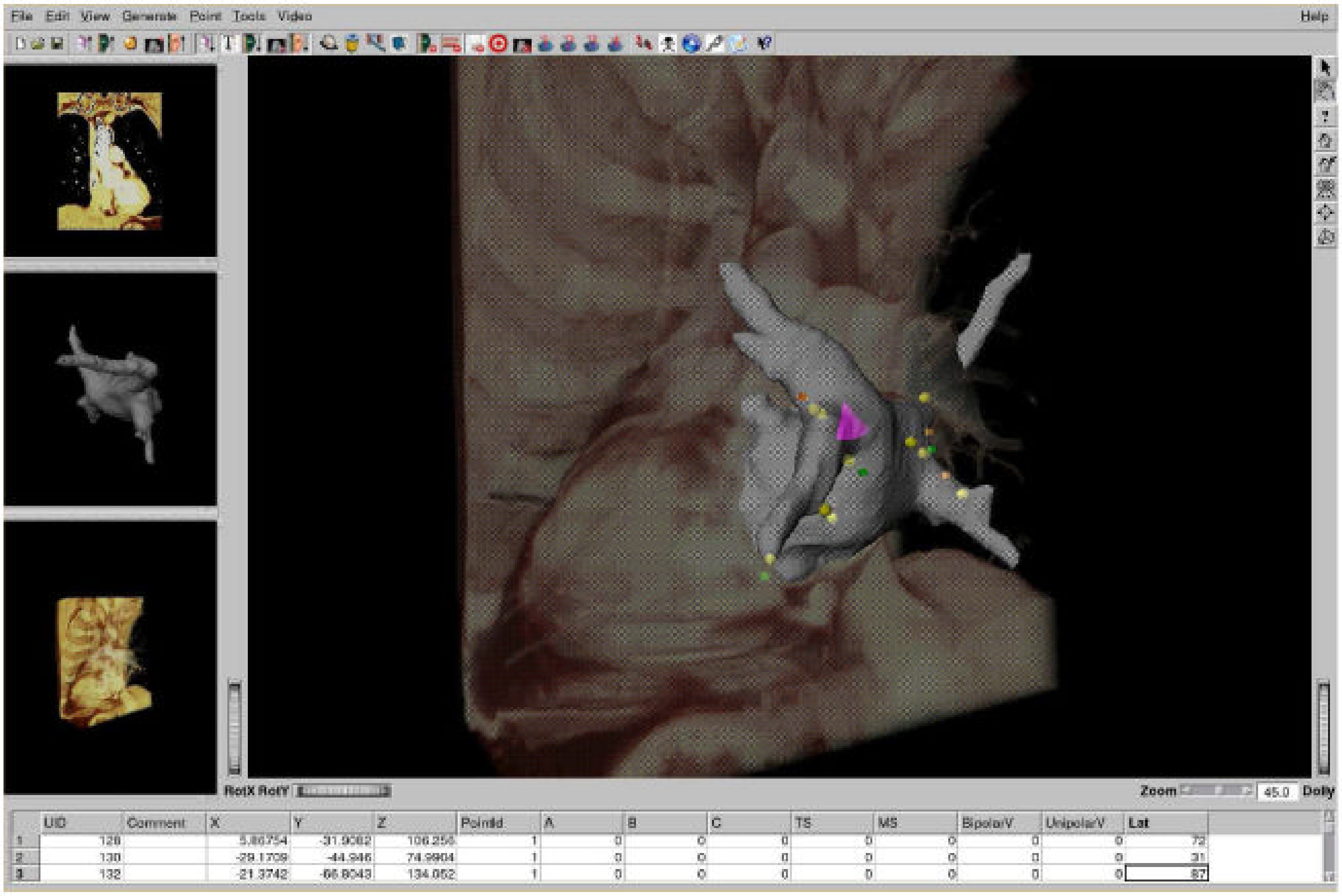


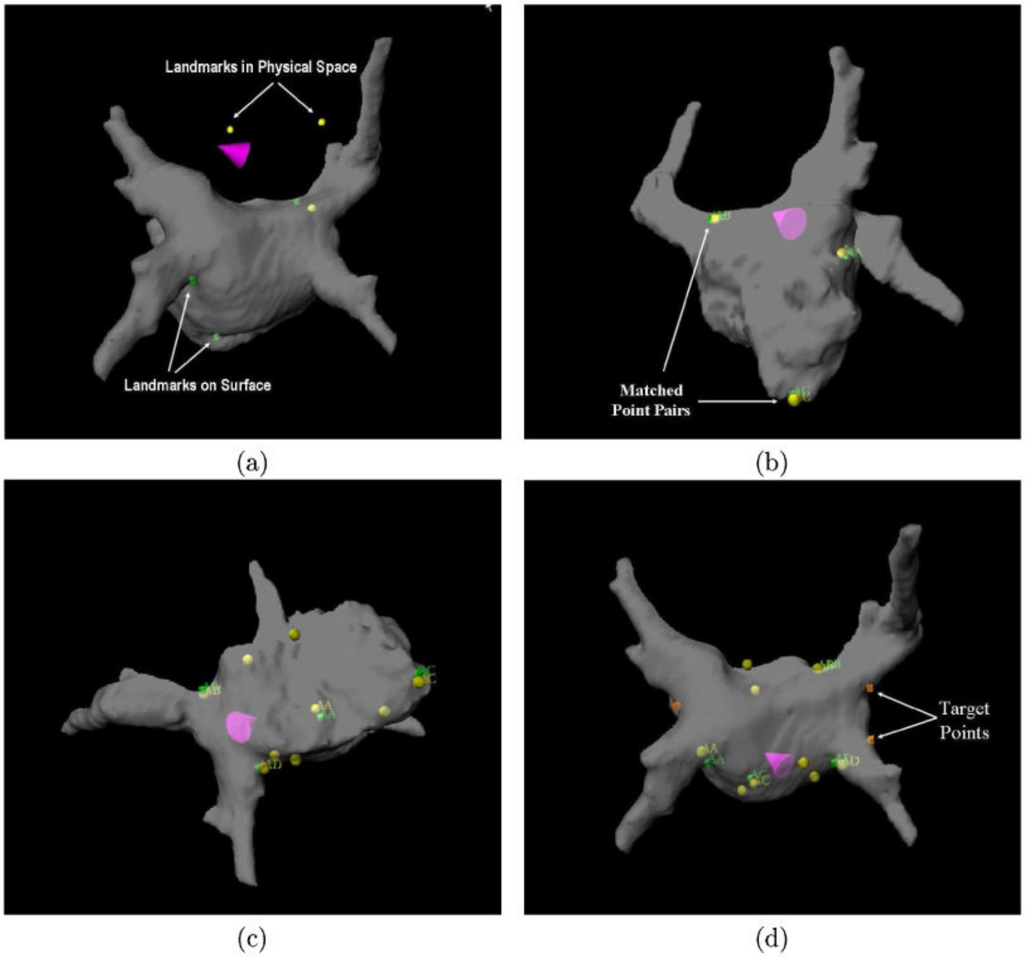
Figure 7.
Photograph of the physical system. The prototype system developed in our laboratory (on right) interfaces to a commercial electrophysiological (EP) mapping system (on left) via an ethernet switch. The ablation switch, attached to the prototype system, is turned on during application of RF energy to track ablation points.



	UID	Comment	X	Y	Z	PointId	A	B	C	TS	MS	BipolarV	UnipolarV	Lat
1	128		5.86754	-31.9082	106.256	1	0	0	0	0	0	0	0	72
2	130		-29.1709	-44.946	74.9904	1	0	0	0	0	0	0	0	31
3	132		-21.3742	-66.8043	134.052	1	0	0	0	0	0	0	0	87

Figure 8.

Snapshot of user interface. An integrated display with surface model, volume rendering, and points is shown in main window and individual data displays are shown in panels along the left.


Landmarks in Physical Space
Landmarks on Surface
(a)
Matched Point Pairs
(b)
(c)
Target Points
(d)

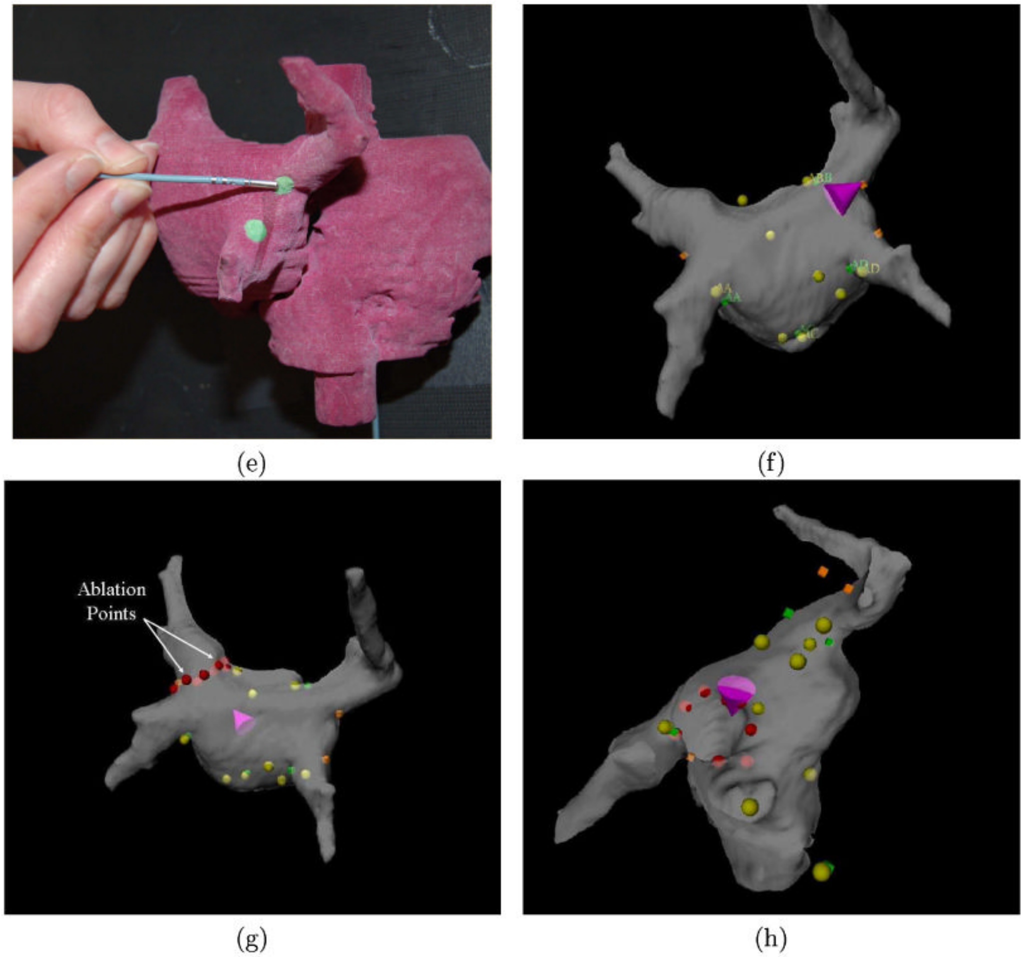


Figure 9.
Emulation of an ablation procedure using the prototype sytem. Landmarks are identified on surface model (green boxes) and in physical space (yellow balls). Points shown (a) before, and (b) after, the landmark-based registration algorithm. Catheter tip is shown as a purple cone. Registration is refined after collection of additional surface points as shown in (c). Target points (d) are displayed as orange cubes. The catheter is placed at the fiducial on the physical model (e), and the corresponding display of the catheter on the surface model is shown in (f). Points collected during an emulated ablation (g) are shown in red. Part of the surface can be “cut-away” to obtain a view of the ablation from inside the left atrium (h).